NOVEMBER-DECEMBER 1976

Technical Supplement ~ 8

FILE UPDATING AND RESTARTABILITY

WORKSPACES AND FILES IN APL WORK ASYNCHRONOUSLY. WHICH CAN CAUSE THE TWO TO BE OUT OF STEP AS A RESULT OF A SYSTEM CRASH.

FOR EXAMPLE. A FUNCTION COULD BE WRITING INFORMATION TO A FILE UP TO THE MOMENT OF A CRASH: BUT THE COPY OF THE WORKSPACE RECOVERED FROM SECONDARY STORAGE COULD BE AT MOST 4 SECONDS OUT OF DATE.

THUS WHEN DESIGNING UPDATING SYSTEMS, IT IS IMPERATIVE TO CONSIDER THE POSSIBILITY OF HAVING FILES AHEAD OF THE FUNCTION, AND TO PERMIT INCORPORATION OF THE NECESSARY CORRECTIVE STEPS INTO THE USER-RESTART ROUTINES.

UPDATING SYSTEMS CAN BE SEPARATED INTO FIVE GENERAL CLASSES:

1) *SIMPLE SEQUENTIAL*
2) *SIMPLE RANDOM*
3) *COMPLEX SEQUENTIAL*
4) *COMPLEX RANDOM*
5) *COMPOSITE*

IN THIS ISSUE WE WILL DISCUSS THE TWO MOST POPULAR UPDATING PROCEDURES - THE 'SIMPLE' UPDATES, SEQUENTIAL AND RANDOM. WE WILL ALSO PRESENT GENERALIZED APL REPRESENTATIONS OF SUCH SYSTEMS, AND USE THEM AS BASES TO DISCUSS THE MORE COMPLICATED PROBLEM OF 'COMPLEX' UPDATES IN FUTURE SUPPLEMENTS.

A SIMPLE SEQUENTIAL PROCESS IS UPDATED BY MEANS OF ⎕APPEND. THE FILE TO BE MODIFIED IS INITIALLY EMPTY AND IS BUILT UP IN SEQUENTIAL STEPS. THE PROCESS IS 'SIMPLE' IF THE INFORMATION PRODUCED FOR EACH COMPONENT CAN BE DERIVED BY ONLY KNOWING THE COMPONENT NUMBER.

THE APL REPRESENTATION IS AS FOLLOWS:

```
     ∇ UPDATESS;I;LIM;OUTFILE
  .
[ ]    I←1
[ ]   L1:(PROCESS I) ⎕APPEND OUTFILE
[ ]     →(LIM≥I←I+1)↑L1
```

'I' IS A MONOTONICALLY INCREASING COUNTER RUNNING TO THE UPPER LIMIT OF PROCESSING STEPS 'LIM'.

'PROCESS' REPRESENTS A 'SIMPLE' COMPUTATION WHICH IS A FUNCTION OF I: EACH EXECUTION OF PROCESS I FOR THE SAME I GIVES THE SAME RESULT.

A SYSTEM CRASH COULD CAUSE THE COUNTER IN THE WORKSPACE TO LAG THE FILE COMPONENTS FOR REASONS DISCUSSED EARLIER. WE WOULD LIKE TO DEVISE A MECHANISM SO THAT, UPON BRANCHING TO A LOCAL RESTART POINT (SAY 'GOON') WE RESET THE NECESSARY COUNTERS.

THE 'RESTARTABLE' FUNCTION WOULD THEN LOOK LIKE THIS:

```
     ∇ UPDATESS;I;LIM;OUTFILE
  .
[ ]    I←1
[ ]   L1:(PROCESS I)⎕APPEND OUTFILE
[ ]   L2:→(LIM≥I←I+1)↑L1
  .
  .
[ ]   GOON:I←(⎕SIZE OUTFILE)[2]-1  ◇ →L2
```

AFTER RESTART, →GOON WOULD RESET THE COUNTER 'I' TO THE LAST COMPLETED PROCESS FOUND IN THE FILE. BRANCHING TO L2 WOULD THEN RESUME THE OPERATION FROM THAT POINT.

THE GENERALIZED SIMPLE SEQUENTIAL PROCESS CAN BE REPRESENTED AS HAVING A FIXED (≥1) NUMBER OF ⎕APPEND'S FOR EACH PROCESSING STEP I. THE CORRESPONDING APL FUNCTION, WITH THE RESTART LINE, WOULD LOOK AS FOLLOWS:

```
     ∇ UPDATEGSS;I;J;N;OUTFILE
  .
[ ]    I←J←1
[ ]   L1:(I PROCESS J) ⎕APPEND OUTFILE
[ ]   L2:→(N≥J←J+1)↑L1
[ ]     J←1 ◇ →(LIM≥I←I+1)↑L1
  .
[ ]   GOON:J←1+N|I←(⎕SIZE OUTFILE)[2]-1 ◇ I←1+⌊I÷N ◇ →L2
```

THE 'PROCESS' FOR EACH I CAN NOW BE VISUALIZED AS CONSISTING OF N SUB-PROCESSES RESULTING IN A TOTAL OF N×LIM ⎕APPEND'S.

SIMPLE RANDOM UPDATES CONSIST OF ⎕REPLACE PROCESSES, WHICH CAN BE CHARACTERIZED AS FOLLOWS:

```
     ∇ UPDATESR;I;LIM;OUTFILE
 .
[ ]    I←1
[ ]   L1:(PROCESS I) ⎕REPLACE OUTFILE, COMP I
[ ]    →(LIM≥I←I+1)↑L1
```

'I' IS A MONOTONICALLY INCREASING COUNTER RUNNING TO THE UPPER LIMIT OF PROCESSING STEPS 'LIM'.

UNLIKE ⎕APPEND, WE HAVE TO SPECIFY AN OUTPUT LOCATION FOR EACH STEP. THIS IS DONE THROUGH THE FUNCTION 'COMP'.

FOR THE PROCESS TO BE 'SIMPLE' WE FURTHER REQUIRE 'COMP' AND 'PROCESS' TO BE FUNCTIONS OF I. EACH EXECUTION OF 'COMP I' AND 'PROCESS I' FOR THE SAME I MUST GIVE THE SAME RESULTS, RESPECTIVELY.

UPON RESTART, WE CANNOT SIMPLY LOOK AT THE END OF THE OUTPUT FILE TO RESET THE COUNTERS. INSTEAD, IN ORDER TO AVOID REPEATING ⎕REPLACES FROM THE POINT OF INTERRUPTION, WE CAN START AT THE CURRENT WORKSPACE LOCATION AND MOVE STEP-BY-STEP UNTIL WE GET A NON-INCREASING ⎕RDCI IN THE OUTPUT FILE. THIS WILL INDICATE AN INTERRUPTION IN OUR SEQUENCE OF UPDATES.

REPRESENTING THE PROGRAM IN APL, WE GET THE FOLLOWING:

```
     ∇ UPDATESR;I;LIM;OUTFILE;T1;T2
 .
[ ]    I←1
[ ]   L1:(PROCESS I) ⎕REPLACE OUTFILE, COMP I
[ ]   L2:→(LIM≥I←I+1)↑L1
[ ]   L3:
 .
[ ]   GOON: →(I∊1,LIM)↑L1 ⍝ I=1: NO TIMESTAMP FOR TESTING IS GUARANTEED
[ ]    ⍝ I=LIM: THERE IS NO SUCCESSOR TO THE LAST ITERATION.
[ ]    T1←(⎕RDCI OUTFILE, COMP I)[3]
[ ]   L99:→(LIM<I←I+1)↑L3
[ ]    T2←(⎕RDCI OUTFILE, COMP I)[3]
[ ]    →(T2<T1)↑L1 ◇ T1←T2 ◇ →L99
```

ALL OF THE ABOVE EXAMPLES ASSUME THAT
(1) NO FILE COMPONENTS WERE DAMAGED,
(2) 1 ORIGIN WAS USED, AND
(3) THE OUTPUT FILE WAS EXCLUSIVELY TIED.

SUMMARY

WORKSPACES CAN BE AS MUCH AS FOUR SECONDS BEHIND FILES AS A RESULT OF A SYSTEM CRASH.
THIS CAN HAVE PARTICULARLY DISASTROUS EFFECTS ON UPDATING MECHANISMS.
TWO METHODS OF RESOLVING INTERRUPTIONS ARE:
1) TO RESTART UPDATING FROM THE BEGINNING (IF THIS IS POSSIBLE), OR
2) TO INCORPORATE, IN ADDITION TO THE RESTART MECHANISM, SOME 'SYNCHRONIZATION' LOGIC.

TEMPUS FUGIT FOR THE TEMPIS FUGIT CONTEST!

THE DEADLINE FOR SUBMISSIONS IS DECEMBER 31. ENTRIES WILL BE JUDGED
FIRST FOR COMPLETENESS, THEN ELEGANCE,
AND FINALLY [ONLY IF NECESSARY] ON SPEED.

A PRIZE WILL BE GIVEN FOR THE BEST OVER-ALL ENTRY. IF THE WINNER
TURNS OUT TO BE AN IPSA-ENTRY, ANOTHER WILL BE AWARDED FOR THE BEST
CUSTOMER ENTRY.

MAIL/BOX ALL ENTRIES TO:

HAL CARIM [MAILBOX CODE: HCA]
I.P. SHARP ASSOCIATES,
145 KING STREET WEST,
SUITE 1400,
TORONTO, ONTARIO,
M5H 1J8
CANADA